Introduction au calcul scientifique avec Matlab

TP2

Correction

Exercice 1.

Observer l'effet des instructions suivantes.

>> x=rand(1,5) => crée un vecteur ligne de 5 termes de valeurs aléatoires comprises entre 0 et 1

>> mean(x) => calcule la valeur moyenne des termes du vecteur x

>> std(x) => calcule l'écart type des termes du vecteur x: std(x) = $\sqrt{\frac{\sum_{i=1}^{5} (xi - mean(x))^2}{N}}$

>> median(x) => retourne la valeur médiane des termes de x: si $x_1<x_2<x_3<x_4<x_5$ alors x_3 est la valeur médiane

>> sort(x) => trie les termes de x du plus petit au plus grand

>> A=rand(3) => crée une matrice 3x3 par des valeurs aléatoires comprises entre 0 et 1

>> sort(A) => trie chaque colonne de la matrice A de manière ascendante (du plus petit au plus grand)

>> [B, I]=sort(A) => retourne dans la matrice B la matrice A triée, et la matrice I contient des indices qui spécifie comment le terme ij de A a été réarrangé dans B : for j = 1:n, B(:,j) = A(I(:,j),j)

>> sort(A') => voir explication sort(A)

>> max(A) => retourne un vecteur qui contient la valeur maximum de chaque colonne de la matrice A

>> max(A') => retourne un vecteur qui contient la valeur maximum de chaque ligne de la matrice A

>> max(max(A)) => retourne la valeur maximum de tous les termes de la matrice A

>> sum(A) => retourne un vecteur contenant la somme des termes de chaque colonne de la matrice A

>> cumsum(A) => retourne une matrice de la même taille que A où chaque colonne est une somme cumulative de la colonne de A : si B= cumsum(A), alors pour chaque colonne j ; $B_{(i+1)j} = A_{(i+1)j} + A_{ij}$ pour i=1, 2

>> prod(A) => rerourne un vecteur contenant le produit des termes de chaque colonne de A

>> diff(A) => retourne une matrice 2x3 don't chaque colonne correspond au vecteur difference de la colonne de A: [A(2:n,:) - A(1:n-1,:)]

>> D=A([1,2],1:3) => D contient la sous matrice de A contenant les lignes 1 et 2 et les 3 colonnes
>> sum(D,1) => retourne un vecteur colonne dont les termes sont la somme des termes de chaque colonne de D

>> sum(D,2) => retourne un vecteur ligne dont les termes sont la somme des termes de chaque ligne de D

Exercice 2.

Soit X une matrice 2xn contenant les coordonnées de n points du plan. Comment faire pour obtenir une matrice où les points sont ordonnés par ordre croissant des abscisses ? (sans utiliser les boucles)

Réponse :

>> B = sortrows(X')' => la matrice B à la même dimension que A et contient les coordonnées des n points avec des abscisses croisants

En effet, la fonction sortarrows permet de trier les colonnes d'une matrice comme un groupe. Vu qu'on souhaite trier les lignes de A alors en fait >> sortrows(A') le résultat est une matrice nx2. Enfin, transpose le résultat pour obtenir une matrice 2xn

Exemple pour n=10 :

>> X = rand(2,10)

X =

0.0971	0.6948	0.9502	0.4387	0.7655	0.1869	0.4456	0.7094	0.2760	0.6551
0.8235	0.3171	0.0344	0.3816	0.7952	0.4898	0.6463	0.7547	0.6797	0.1626

>> sortrows(X')

ans =

0.0971	0.8235
0.1869	0.4898
0.2760	0.6797
0.4387	0.3816
0.4456	0.6463
0.6551	0.1626
0.6948	0.3171
0.7094	0.7547
0.7655	0.7952
0.9502	0.0344

>> sortrows(X')'

ans =

0.0971	0.1869	0.2760	0.4387	0.4456	0.6551	0.6948	0.7094	0.7655	0.9502
0.8235	0.4898	0.6797	0.3816	0.6463	0.1626	0.3171	0.7547	0.7952	0.0344

Remarque : Pour trouver la fonction « sortarrows » il faut regarder les fonctions qui ont une relation avec la fonction principale qui est « sort ». Ces fonctions sont toujours proposer par le help de la fonction principale

Exercice 3.

1. Soit le vecteur de dimension 8 de composantes : 3.2; 4.8; 3.3; 3.2; 3.1; 4.2; 3.2; 3.3 Construisez le vecteur y = $(y_i)_{i=1;...;8}$ correspondant.
2. Construisez, à l'aide des fonctions vectorielles (sans utiliser les boucles), une fonction *moyenne.m* qui a comme argument d'entrée n et qui permet de calculer la suite des moyennes mobiles:

$$\bar{y}_n = \frac{1}{n}\sum_{i=1}^{n} y_i \ .$$

3. Donner la valeur de $\bar{y}_8$.

Réponse :

1. >> y= [3.2; 4.8;3.3; 3.2; 3.1; 4.2; 3.2; 3.3]

2. - Créer un fichier moyenne.m
   - Y mettre le code suivant :

```
function [yn] = moyenne(n)
%MOYENNE calcule le terme n de la suite des moyennes mobiles du vecteur y
%   vecteur y
y= [3.2; 4.8;3.3; 3.2; 3.1; 4.2; 3.2; 3.3];

yn = sum(y(1:n))/n;
end
```

3. $\bar{y}_8$ = sum(y(1:8)) = 3.5375

Remarque : étant donné que y contient 8 terme alors $\bar{y}_8$ est finalement la moyenne de y, donc on peut aussi retrouver le même résultat en faisant >> $\bar{y}_8$=mean(y)

Exercice 4.

Soit la matrice A :

```
A=[1  2  3;  2  3  1;  3  1  2]
```

Construisez, à l'aide des opérateurs logiques, la fonction *find.m* qui a pour argument d'entrée un scalaire k et qui permet de retourner les termes égaux à k ainsi que leur position.

Réponse :

Etant donné de « find » est déjà une fonction de matlab, et afin d'éviter tout conflit, on crée une fonction appelée « find_A »

- Créer un fichier find_A.m
- Y mettre le code suivant :

```
function [I,J] = find_A(k)
%FIND_A retourne les positions ij des termes égaux à k de la matrice A
% I est le vecteur des indices i et J est le vecteur des indices j
% la matrice A
A=[1 2 3; 2 3 1 ; 3 1 2];
%initialisation
I=zeros(3,1);
J=zeros(3,1);
e=1;
l=1;
for i=1:3
   for j=1:3
      if A(i,j)==k
         I(e)= i;
         J(l)= j;
         e=e+1;
         l=l+1;
      end
   end
end
```

Pour vérifier que ma fonction est juste, je vais comparer son résultat au résultat de la fonction « find » de Matlab.

Exemple pour k= 2:

```
>> [I,J]=find_A(2)

I =

     1
     2
     3


J =

     2
     1
```

```
     3

>> [J,I]=find(A==2)

J =

     2
     1
     3

I =

     1
     2
     3
```

On retrouve bien les mêmes positions des termes égaux à 2, la seule différence est que le premier argument de ma fonction find_A est le vecteur des indices i et le deuxième argument et celui des indices j. Contrairement à la fonction « find » de Matlab pour laquelle le premier argument est le vecteur des indices j et le deuxième est celui des indices i.

Remarque : Il faut toujours bien lire les notices des fonctions Matlab pour mieux comprendre et bien utiliser les arguments d'entrée et de sortie de ces fonctions.

Exercice 5.

Reprenez le vecteur y de l'exercice 3.

1. Ecrire une fonction *existe.m* qui a comme argument un vecteur quelconque z et un scalaire k et qui permet de retourner la réponse à la question suivante : existe-t-il une coordonnée du vecteur z inférieur à k.
2. Utiliser la fonction existe.m pour répondre à la question suivante : existe-t-il une coordonnée du vecteur y inférieure à 3.3

3. Ecrire un script qui permet de construire un vecteur logique z tel que la ième coordonnée de ce vecteur sera 1 si la ième coordonnée du vecteur y est à l'extérieur de l'intervalle :

$$[\bar{y}_8 - \sigma_8, \bar{y}_8 + \sigma_8]$$

Où σ_8 est la racine carrée de la variance d'échantillonnage.

Réponse :

1. - Créer un fichier existe.m
   - Y mettre le code suivant :

```
-----------------------------------------------------------------------------------------------------------
function [r] = existe(z,k)
%EXISTE permet de répondre à la question existe-t-il une coordonnée du vecteur z inférieur à k
n=size(z);
r='non';
for i=1:n
```

```
    if z(i) < k
        r = 'oui';
    end
end
end
```

---

2. Exemple pour z=y, k=3.3 et k=1 :

```
>> r=existe(y,3.3)

r =
    'oui'

>> r=existe(y,1)

r =
    'non'
```

3. –Créer un fichier script appelé ici logic.m
   - Y mettre le code suivant :

```
%initialisation des variables
y= [3.2; 4.8;3.3; 3.2; 3.1; 4.2; 3.2; 3.3];
n= size(y);
z = zeros(n);
% la racine carré de la variance est égale à l'écart type
sigma = std(y);
% la moyenne de y
ym = mean(y) ;
% les valeurs de l'intervale
inf = ym-sigma;
sup = ym + sigma;
for i=1:n
    if (inf <= y(i)) && (y(i) <= sup)
        z(i) = 1;
    end
end
```

En exécutant logic dans la console Matlab et en affichant z, on obtient :

```
>> logic
>> z
z =
     1
     0
     1
     1
     1
     0
     1
     1
```